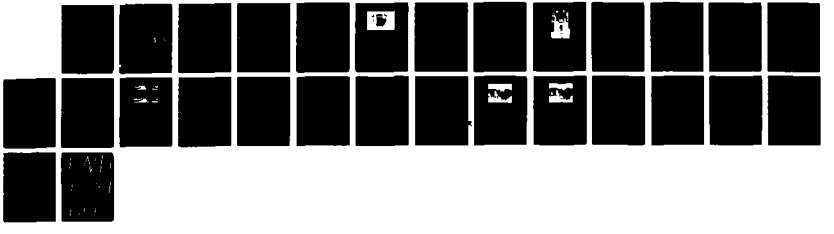
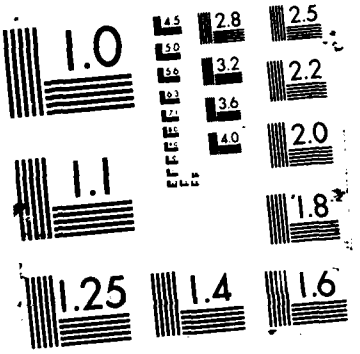
AD-A185 602 SELF CALIBRATION OF MOTION AND STEREO VISION FOR MOBILE ROBOT NAVIGATION(U) MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE L.. 1/1

UNCLASSIFIED R A BROOKS ET AL. AUG 87 AI-M-984 F/G 12/9 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(12)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER: AI Memo 984	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4 TITLE (and Subtitle): Self Calibration of Motion and Stereo Vision for Mobile Robot Navigation		5. TYPE OF REPORT & PERIOD COVERED: memorandum
		6. PERFORMING ORG. REPORT NUMBER
7 AUTHOR(s): Rodney A. Brooks, Anita Flynn, Thomas Marill		8. CONTRACT OR GRANT NUMBER(s): N00014-86-K-0685 N00014-85-K-0124
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE: August 1987 13. NUMBER OF PAGES: 25
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report): UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Mobile robots
Self calibration
Motion vision
Stereo vision

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We report on experiments with a mobile robot using one vision process (forward motion vision) to calibrate another (stereo vision) without resorting to any external units of measurement. Both are calibrated to a velocity dependent coordinate system which is natural to the task of obstacle avoidance. The foundations of these algorithms, in a world of perfect measurement, are quite elementary. The contribution of this work is to make them noise tolerant while remaining simple computationally. Both the

over

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20 cont.

algorithms and the calibration procedure are easy to implement and have shallow computational depth, making them (1) run at reasonable speed on moderate uni-processors, (2) appear practical to run continuously, maintaining an up-to-the-second calibration on a mobile robot, and (3) appear to be good candidates for massively parallel implementations.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A. I. Memo 984 August, 1987

SELF CALIBRATION OF MOTION AND STEREO VISION FOR MOBILE ROBOT NAVIGATION

Rodney A. Brooks, Anita M. Flynn and Thomas Marill

Abstract. We report on experiments with a mobile robot using one vision process (forward motion vision) to calibrate another (stereo vision) without resorting to any external units of measurement. Both are calibrated to a velocity dependent coordinate system which is natural to the task of obstacle avoidance. The foundations of these algorithms, in a world of perfect measurement, are quite elementary. The contribution of this work is to make them noise tolerant while remaining simple computationally. Both the algorithms and the calibration procedure are easy to implement and have shallow computational depth, making them (1) run at reasonable speed on moderate uni-processors, (2) appear practical to run continuously, maintaining an up-to-the-second calibration on a mobile robot, and (3) appear to be good candidates for massively parallel implementations.

Acknowledgments. This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the research is provided in part by the University Research Initiative under Office of Naval Research contract N00014-86-K-0685, in part by an IBM Faculty Development Award, in part by a grant from the Systems Development Foundation, and in part by the Advanced Research Projects Agency under Office of Naval Research contract N00014-85-K-0124.

1. Introduction

This paper is about vision for a mobile robot. It is about how to build a computationally cheap robust vision system which delivers data for obstacle avoidance. The vision system is continually self calibrating, making it tolerant of normal mechanical drift. But better than that, it is tolerant of severe blows to its head-like sensor platforms. After a few seconds in a trauma-induced daze it adapts to its grossly altered sensor alignments.

But wait, there's more! The algorithms require no pre-knowledge of camera focal lengths, fine orientation, or stereo baseline separation. With such quick calibration and adaption the sensors can be mounted on cheap steerable systems. We can trade cheap computation for deficiencies which arise from avoiding expensive mechanical solutions to sensor steering problems.

We begin with three observations:

- Humans are able to extract meanginful information from images without being aware of the camera geometry (e.g. baseline separation in stereo) or the focal parameters of the imaging system. They can adapt almost instantly to TV or movie images made with unknown optics and relatively quickly to disturbances in the optical pathway to their retinas. For instance a change in stereo baseline separation induced by special glasses can be adapted for in a matter of seconds. In contrast, most mobile robots today require precisely understood optics and many seconds of intense computation at the begining of an experimental run (Faugeras and Toscani (1986)) to accommodate small mechanical and electronic drifts in their systems.
- Marr (1982) points out that the *purpose of vision* depends on the task the perceiving organism is trying to achieve. Brooks (1986) demonstrates that data from a single sensor system can be used in entirely independent channels (including independent perception systems) to control different aspects of the behavior of a mobile robot. Thus there need not be just a single purpose of vision. A useful engineering consideration in building a perception system, then, is to analyze the requirements in terms of the task to be achieved using the output of the perception system.
- Much of human and animal vision is extremely fast, taking place in small fractions of a second. However, it is implemented on hardware which is extremely slow (perhaps one thousand gate delays per second) as compared with today's computer hardware (ten to one hundred million or more gate delays per second). Nonetheless biological vision is vastly superior to current computer implementations. Biological algorithms must therefore be computationally shallow in order to fit on the available hardware.

In this paper we explore some visual techniques useful for a mobile robot navigating in indoor environments. The particular task these techniques support is avoiding obstacles. Brooks (1986) shows how to separate this particular task from others that a mobile robot may concurrently be pursuing.

Noting the earlier observations, we are interested in finding self calibrating algorithms which are tolerant of large drifts in optical properties of the imaging system, in finding algorithms which have shallow computation depth, and in finding algorithms that are well suited to the obstacle aviodance task.

The experiments we describe in this paper have been performed using Allen the robot (Brooks (1987)) as a sensor platform and an offboard Lisp machine as the computational

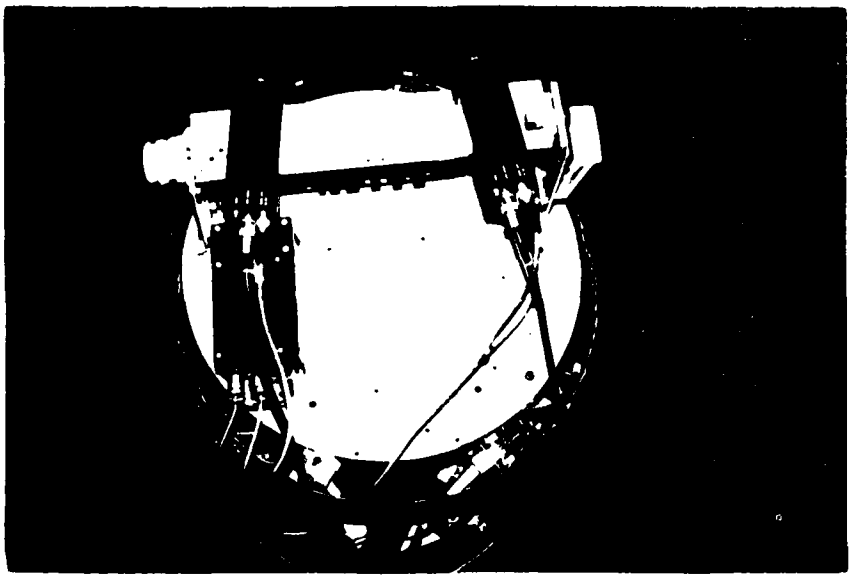


Figure 1. Two cameras are mounted on Allen using standard tripod mounts. With such mounts it is certain that the cameras will not be parallel. There is also mechanical misalignment between the camera mounts and the drive mechanism of the robot. A more expensive mechanical system, and a more complex camera mounting system could overcome these limitations, but the system would still be susceptible to misalignment through minor mechanical damage. If we don't demand mechanical alignment we won't be perturbed by normal wear and tear.

engine. The robot has two approximately forward-looking CCD cameras. A standard camera mount is used to attach them to a tilt head, so there is considerable risk of not having the cameras pointing directly forward. Figure 1 shows the cameras mounted and on the robot.

1.1 Forward motion and stereo

Our primary algorithm analyzes forward motion by tracking strong vertical edges over many consecutive images. As Bolles, Baker and Marimont (1987) have pointed out, one avoids the problem of solving for corresponding points by taking closely spaced images.

Unlike Bolles, Baker and Marimont, however, we are addressing the problem of forward rather than lateral motion, and we do not assume knowledge of camera motion or of camera angle relative to motion.

Forward motion analysis relies on straight line motion of the robot at constant, but unknown, velocity, but does not require that the camera point directly ahead, nor that it point at a known angle relative to the motion. There are strong constraints on the possible motion of features if the robot motion constraints are met, so it is often possible to detect when the motion constraints are being violated.

The particular way in which forward motion analysis supports the obstacle avoidance task is to deliver distances to obstacles in units of time to collision at the current velocity. This is the perfect coordinate system for obstacle avoidance. Steering commands and velocity change commands can be given without the need to convert from an absolute coordinate system to desired velocities for the robot.

Forward motion analysis has some surprising independence properties;

- the details of the focal geometry of a perspective camera are unimportant,
- the camera need not be pointing directly in the direction of motion (there is a simple procedure which recovers the orientation of the camera relative to the direction of motion),
- and it naturally delivers the distance to the physical artifacts giving rise to tracked features as the time to collision with that artifact in units of the inter-frame time intervals.

These properties make forward motion analysis ideal as an input to an obstacle avoidance task on the robot. Additionally it is easily implementable on hardware which supports only 16 bit arithmetic.

Unfortunately forward motion analysis has some drawbacks also;

- it doesn't deliver distances to obstacles which are straight ahead along the direction of motion (and clearly these are some of the most important obstacles to consider),
- and it is useless when the robot is stationary or turning in place. Thus when the robot stops to turn, it is completely blind in its new direction of motion until it has moved in that direction for a period of time. In a cluttered environment it may be important to be able to plan an obstacle free path before starting to move.

To compensate for these shortcomings we use a second early vision algorithm whose properties exactly complement those of forward motion analysis. We use a single scanline stereo algorithm. Stereo provides depth measurements

- straight ahead,
- and when the robot is stationary.

But stereo too has a serious drawback. Even rough depth measurements rely on a knowledge of camera focal geometry and the six parameters relating the geometries of the two cameras. Alignment errors as small as $\pm 5°$ for each camera can lead to wildly incorrect depth estimates, including negative depths and an order of magnitude wrong positive depths.

The main result of this paper is a simple method of continously calibrating the stereo system to reliably deliver depths in the same units as the forward motion analysis algorithms.

1.2 Experimental assumptions

Man-made indoor environments give rise to images with strong vertical edges. We can make use of this fact in designing our algorithms.

The forward motion and stereo algorithms both assume one dimensional cameras with the plane defined by the image line and the optical center parallel to the ground (this was also assumed by Bolles, Baker and Marimont (1987)). In the current experiments we use regular video CCD cameras and average a swath of 16 scanlines from the middle of the image. This highlights strong vertical edges. Grey levels are taken to 8 bits and range from 0 (white) to 255 (black).

Two cameras were mounted side by side facing forward, separated by approximately 8 inches. The cameras were only approximately aligned in the forward direction. Our analysis below allows up to a $\pm 5°$ misalignment of each camera. It assumes the cameras are restricted to a field of view of 60°. Knowing the field of view lets us compute the focal ratio of the camera. The analysis assumes the two cameras are identical in this regard but is easily generalized to two different but known cameras.

In our experiments we used cameras and lens systems with approximately a 60° field of view. There is only one place, and that is in the forward motion algorithm, where knowledge of the field of view and hence focal ratio is used. In section 2 we show that in order to relate the depth estimates, for stereo calibration, obtained from two cameras we must include a small error term to correct for their misalignment. It is only in this error term that the field of view is needed. We show that these errors can be at most $\pm 5\%$ so only an approximate knowledge of the field of view is necessary. For a robot which can turn in place and which has appropriate encoders to measure its turn, the camera field of view could easily be calibrated each time it turns by tracking image points over a signifcant portion of the image. We did not so estimate the field of view in these experiments but assumed it *a priori*.

All computation was done offboard and offline on a Symbolics lisp machine. Total computation time for the reported experiments is on the order of a few seconds running unoptimized lisp code.

We plan on porting these algorithms to a new robot with all onboard computation. On that robot we plan on using cylindrical lenses and single scan line horizontal CCD cameras. The lenses will optically average a horizontal swath of a more conventional image highlighting strong vertical edges.

For the task of obstacle avoidance we estimate that knowledge of the distance to obstacles to within $\pm 10\%$ is quite adequate.

All our experiments described in this paper give distances in terms of the robot's current speed. We did not try to measure that quantity. Our evaluation of our experiments cannot therefore be based on determinations of absolute accuracy. Rather, we compare computed relative distances to pairs of tracked objects as a measure of accuracy, as in an ideal experiment all such relative distances should remain constant. On this basis we believe we have achieved $\pm 10\%$ accuracy with very computationally simple and shallow depth algorithms.

2. Forward Motion Vision

In this section we first derive the equations of forward motion vision without regard to sensitivity to noise and sampling errors. We then describe some practical algorithms to overcome problems due to noise and sampling errors.

2.1 One dimensional camera geometry

Consider figure 3. It is a view from above of a camera. The *image plane* is a one dimensional strip, i.e., the camera provides only a 1-D image.

The image plane has P pixels, numbered 0 through $P - 1$. The *optical center* of the camera is distance f, measured in pixels, from the image plane. We call f the *focal ratio*.

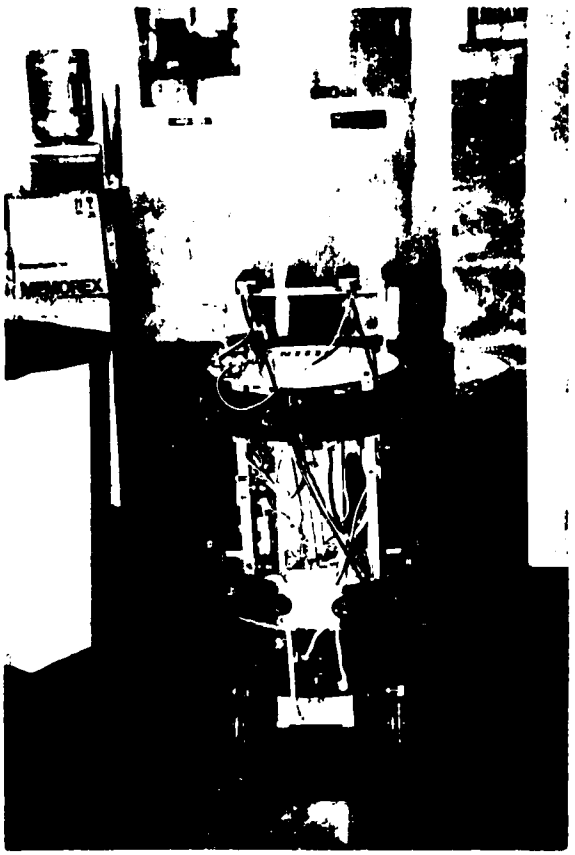


Figure 2. In our experiments Allen ran in a straight line between various obstacles. The scene was deliberately cluttered so that we could get a good calibration with only 100 stereo image pairs. We are currently limited by computer memory from collecting more than that number in real time for later analysis. In future experiments we expect to do all computation onboard and to be able to calibrate over many more images in less cluttered environments.

The *optical center plane* is parallel to the image plane and runs through the optical center. A line, the *optical axis*, runs through the optical center and intersects the image plane at right angles. The point of intersection divides the image plane in two equal parts and is called the *center of view*.

Points in the world are projected onto the image plane along lines that run through the optical center. The *field of view* of the camera is ϕ, the maximum angle subtended by any two visible points. The focal ratio and field of view are related by:

$$f = \frac{P}{2\tan(\phi/2)}. \tag{1}$$

This gives f in units of pixels.

On the cameras we used in our experiments, which are 576 pixels wide with approximately a 60° field of view, f is approximately 499 pixels.

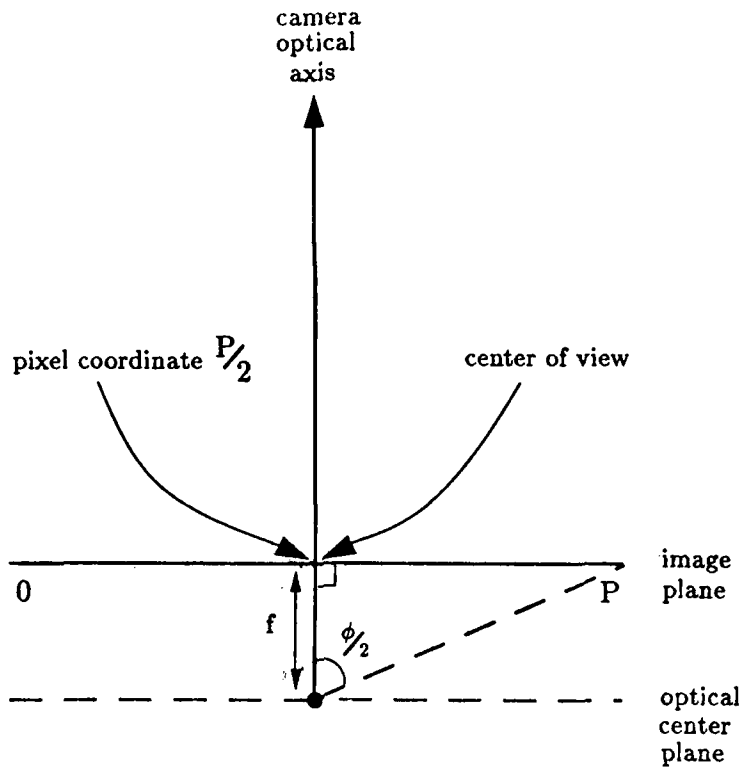


Figure 3. A one dimensional camera has an image plane f pixel units from the optical center of the camera. The optical axis is perpendicular to the image plane, which is P pixels wide.

2.2 Forward motion vision geometry

Now consider figure 4. It is a view from above of a camera whose optical axis is offset somewhat from the direction of motion of the camera. The camera is undergoing pure translation in a constant *direction of motion.* The motion direction vector, the optical axis and the image strip on the image plane are all coplanar.

This model corresponds to a 1-D camera mounted parallel to the ground plane on a mobile robot which is travelling forward without turning. Typically one would mount a cylindrical lens on such a camera to enhance the strong vertical edges found in man-made indoor environments. Without careful alignment of the camera it will not point directly in the direction of motion of the robot. Even if the camera is rigidly mounted on the body of the robot, the direction the robot travels with respect to its body will vary gradually over time due to tire wear and other mechanical processes. On our robot Allen we notice that the body and drive base of the robot can easily be misaligned by a few degrees. In this section we show how to calibrate dynamically for these effects with a few arithmetic operations over a few images taken as the robot is moving at constant velocity.

The trajectories of image features under the geometry of figure 4 are described by Bolles,

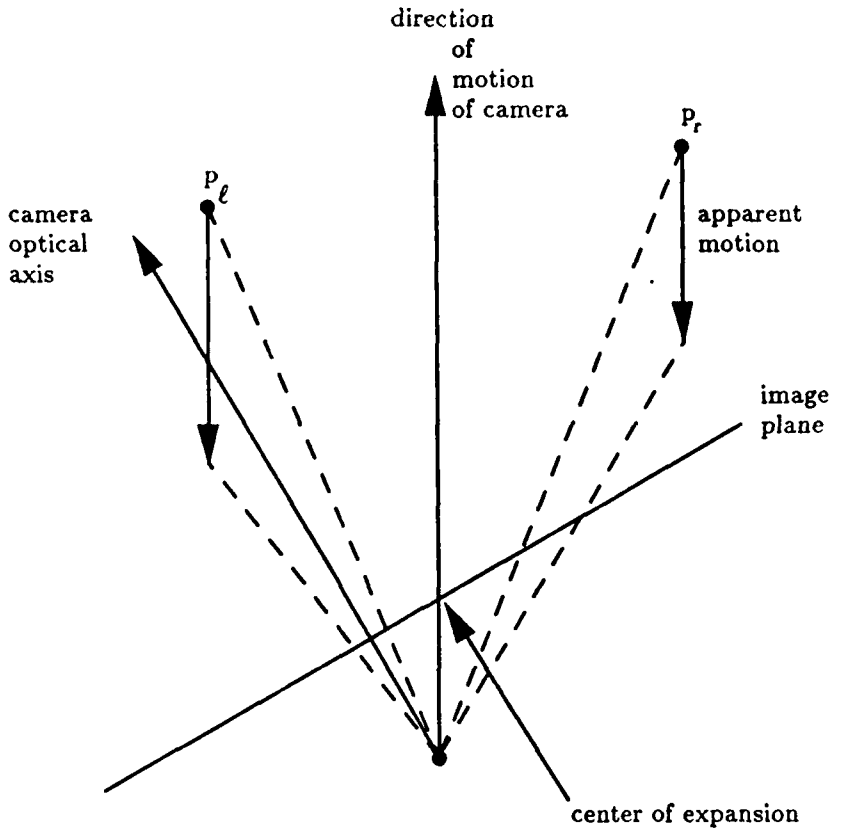


Figure 4. As a camera moves in a straight line, points to the left of the direction of motion of the camera appear to move to the left and those to the right appear to move to the right.

Baker and Marimont (1987). Here we derive equations yielding the time-to-collision and the center of expansion under this geometry.

The image of any point p_r which is to the right of the direction of motion will move rightwards in the image plane. The image of any point p_l which is to the left of the direction of motion will move leftwards in the image plane. Points directly ahead will not move at all in the image. The projection of such points is called the *center of expansion.* We write C_E, for its coordinate in pixels.

In an operational robot, we envision an estimator for C_E running continously and slowly and incrementally updating the estimate as it drifts. More generally what we would like to know, for the purpose of controlling a robot, is how long it will be before the robot reaches some visible point p. The forward motion analysis described here determines the time that will elapse for the robot to travel forward far enough that the plane, parallel to the image plane and coincident with the optical center, intersects point p. This assumes that the robot's velocity is constant but not that it is known.

Now consider figure 5, which illustrates the same physical setup as the last figure but with different quantities annotated. The camera has focal ratio f. Call the distance from the optical center to the center of expansion f'. The camera is misaligned by an angle

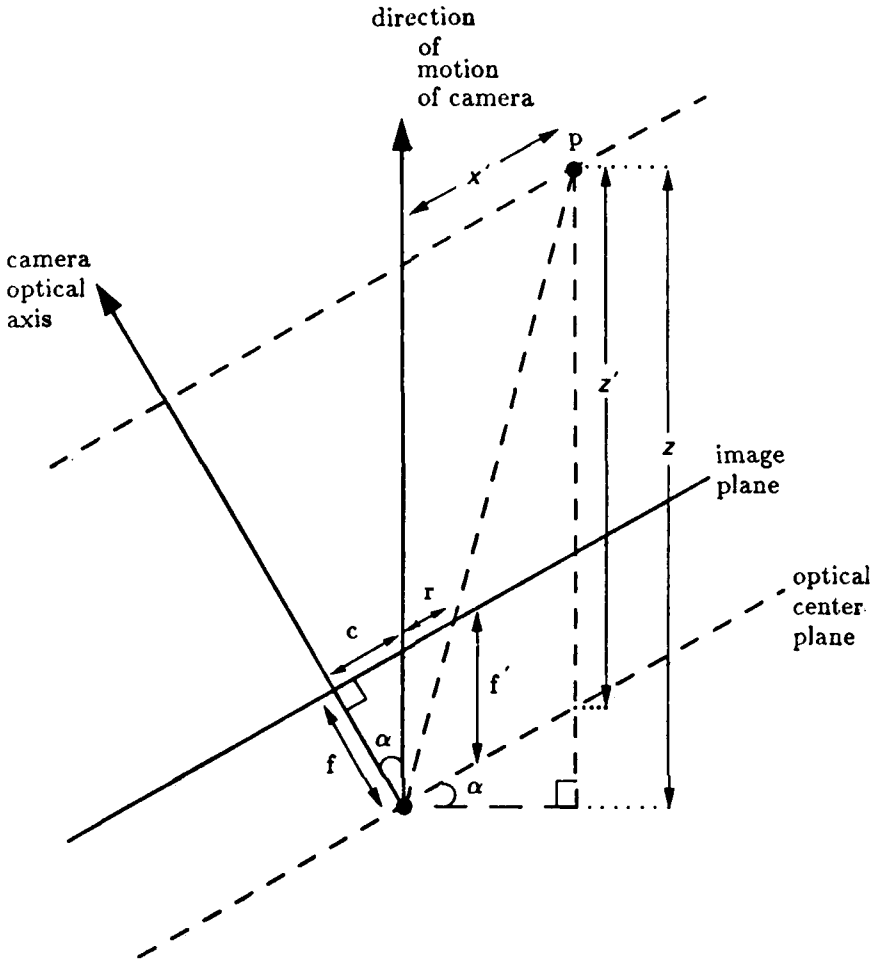


Figure 5. As a camera moves forward in some direction α offset from the optical center of the camera, we are interested in computing the distance z to a point p.

α from pointing directly in the direction of motion. Suppose the camera is moving with constant velocity v and consider what happens to the image of a point p. Its physical location can be described by two non-orthogonal coordinates: x', its distance parallel to the image plane from the path taken by the optical center of the camera, and z', its distance in the direction of motion from the optical center plane. Note that the derivative of z' with respect to time, $\dot{z}'$ is the constant $-v$.

From similar triangles we can see that the distance, r, along the image plane, of the image of point p from the center of expansion is given by:

$$r = \frac{f'x'}{z'} \tag{2}$$

and since x' is constant its derivative with respect to time is:

$$\dot{r} = -\frac{f'x'}{(z')^2}\dot{z}'.$$

We can then observe that:

$$\frac{r}{\dot{r}} = -\frac{z'}{\dot{z}'} = \frac{z'}{v}$$

The last term is the time it will take for the optical center plane to intersect p. Thus $r/\dot{r}$ is just the *time to collision* of the optical center plane and the physical feature whose image is being observed.

The accuracy of forward motion analysis is very sensitive to the estimate we make of the center of expansion since r is measured relative to it. The position of C_E can range over a third of the image when camera alignment of a 30° field of view camera is allowed to vary by ±5° (the range is not so bad for larger field of view cameras, but the quantity being measured is correspondingly smaller). Its value clearly impacts greatly the estimates of time to collision.

With perfect measurement it turns out to be easy to compute the value of C_E by tracking a single point over a small time interval. Suppose we measure the image position of a point in a coordinate frame relative to the left end of the image. It has position R, say, in this coordinate system. Since C_E should remain constant we have that

$$r = R - C_E, \quad \dot{r} = \dot{R}$$

so the time to collision for the optical center plane is

$$\frac{r}{\dot{r}} = \frac{R - C_E.}{\dot{R}} \tag{3}$$

Suppose we can measure R and $\dot{R}$ at two distinct times t_0 and t_1. Then the estimate to time to collision should decrease by precisely $t_1 - t_0$. Writing our estimates as functions of time we then get that:

$$\frac{R(t_0) - C_E}{\dot{R}(t_0)} - (t_1 - t_0) = \frac{R(t_1) - C_E}{\dot{R}(t_1)}$$

whence

$$C_E = \frac{\dot{R}(t_0)R(t_1) - \dot{R}(t_1)R(t_0) + \dot{R}(t_1)\dot{R}(t_0)(t_1 - t_0)}{\dot{R}(t_0) - \dot{R}(t_1)} \tag{4}$$

In summary then we can easily (given perfect measurement) estimate C_E and therefore compute the time to collision of the optical center plane

$$\frac{z'}{v} = \frac{R - C_E}{\dot{R}}. \tag{5}$$

However, time to collision from the optical center plane to the point p, i.e., z'/v is not exactly the information we want if the camera has been misaligned. In order to calibrate our stereo system in section 3 of this paper we will need depth estimates in a common coordinate system for the two cameras. Therefore we really want z/v where z is the component of distance from the center of the camera in the direction of motion and

$$z = z' + x' \sin\alpha \tag{6}$$

as can be seen from figure 5.

Let $c = C_E - P/2$, the distance in pixels along the image plane from the center of view to the center of expansion, then referring to figure 5

$$\sin\alpha = \frac{c}{f'}$$

so by equation (2)

$$z = z' + \frac{cx'}{f'} = z' + \frac{crz'}{(f')^2}$$

and since

$$f' = \sqrt{c^2 + f^2}$$

we have

$$z = z' + \frac{crz'}{c^2 + f^2}. \tag{7}$$

Notice that since we only really know z'/v (from equation (5)) this only gives us z/v, which in any case was what we wanted.

It is interesting to ask just how different z can be from z'. Since α is restricted to $\pm 5°$, x' is almost perpendicular to z', so for a camera with a 60° field of view

$$-\tan 30° \leq \frac{x'}{z'} \leq \tan 30°$$

and since $\tan 30° \times \sin 5° = 0.0503$ we see that $z = z' + x' \sin\alpha$ is within $\pm 5\%$ of z'. Thus, since equation (7) is correcting such a relatively small error it is not critical to know f (which can be derived from ϕ through equation (1)) particularly accurately.

We now know z/v, or the *time to collision* with the obstacle, very accurately using only an imprecise approximation to the field of view of the camera as our *a priori* knowledge (which is easily gained by rotating the camera a roughly known amount). Since time to collision is precisely the right quantity to know for obstacle aviodance we don't need to try to compute the robot's velocity v at all. In fact we don't even need to know how our measure of passage of time t relates to external units. All we need is a steady on board clock and we can do obstacle aviodance visually.

2.3 Tracking image features

The above analysis suggests the idea of detecting image features and tracking them over many images.

We track edges. We simply convolve the image with a derivative of a Gaussian (the actual mask is 1, 3, 5, 9, 14, 18, 20, 18, 11, 0, −11, −18, −20, −18, −14, −9, −5, −3, −1) and then look for local maximum absolute values (Horn (1986)). We threshold the edges based on their strength of convolution with the mask. We accept only edges which have a convolution value greater than 500. We do not attempt sub-pixel localization of edges as our robot shakes enough as it rolls foward to impose at least ±1 pixel error on top of any localization noise due to discrete estimates.

Figure 8 is a data set showing edge traces from a run with the mobile robot as the robot moves forward. It is a 2-D binary array where each row is a one dimensional image and time flows downwards. Array elements are 1 (black) where an edge was detected and 0 (white) otherwise. Equation (2) and the constant velocity of the robot tell us that each edge trace is a hyperbola. If the images are taken sufficiently close together (such as in this dataset) it is very simple to track edges without having to refer to the original grey level images.

It is sufficient to buffer only two rows of the array and keep track of the direction and velocity of an edge track in order to predict a small search window and direction of search in the second row. There are two cases in searching for the next edge in a trace; at the beginning of an edge trace where the direction within the image may not be known, and

later when the direction is known. At all times the possibility of noise in the edge motion must be taken into account.

When the direction is not known a default symmetric window (± 3 pixels) is searched in a succeeding row. If only one edge element is found, that is taken to be the next element of the trace. If more than one edge is found the current trace is abandoned.

When the direction of motion (left or right) is known we keep track of the edge velocity at each step of the trace. The trace is predicted to move the same amount at the next step plus or minus a small window of margin in order to accomodate noise in the image and edge velocity increases; -1 pixel short of prediction through $+3$ pixels extra. The search proceeds in the direction of motion, and the first edge encountered is taken as the correct edge. If no edge is encountered within the window then the edge trace is abandoned. No attempt is made to hypothesize a missing edge in the given one dimensional image and to try to look in the appropriate place in the next image.

There are two ways in which the direction of edge motion can be determined.

After a few elements of a trace have been tracked the direction should be clear by comparing the position of the first and most recent pixels. If the direction is not clear then the trace can be abandoned as there is no obvious depth information in it.

However, most edge traces have only one possible direction even considering only the first row in which it appears. If C_E is known then edges on the left move leftwards, and edges on the right move rightwards. Even when C_E is not known, an a priori knowledge of f and the maximum permissible range of α determines a strip within the image where it is possible that C_E might fall, and edge traces starting outside that strip have the obvious direction.

The key heuristic used by the above edge tracking algorithm to handle noise is perhaps not obvious at first glance. In fact the main idea is to abandon an edge trace when conditions get complicated. Chances are that the disturbance will last only a few images at most and the trace can be re-established as a brand new trace a few images later.

2.4 Noise and estimation

The analysis of section 2.1 assumes no noise and perfect measurement of various quantities. Real image sequences suffer from digitization effects and large sources of noise due to unstable camera platforms, unconstant velocity and curved rather than straight line motion. We must therefore fit all our measurements over many images.

To estimate the center of expansion we need to know R and $\dot{R}$ at more than one place along an edge trace. To decrease the effects of noise it is clearly best to use more than one edge trace. We use every edge trace and continually refine our estimate of C_E.

Quantity R is measured directly from the edge pixel array. To estimate $\dot{R}$, the edge velocity, we trade off localization in time of our estimate with accuracy of the estimate through the choice of a constant V. In all our experiments reported here we have used $V = 4$. We approximate $\dot{R}$ by the slope of a chord between two points on a hyperbolic trace V images apart. By Rolle's theorem some point on the hyperbola between those two points has exactly that slope. We arbitrarily choose the midpoint in time. A larger V reduces the magnitude effects of noise on the estimate but increases our localization error. We see that $V\dot{R}$ is simply the distance travelled by the edge trace over V time intervals.

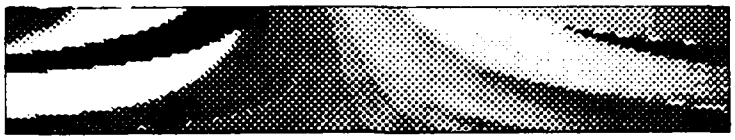


Figure 6. Image strips averaged to a single scan line and ordered in time for 100 images from the left camera.

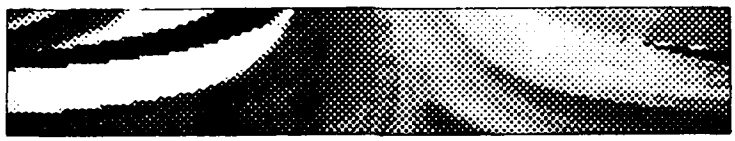


Figure 7. Image strips averaged to a single scan line and ordered in time for 100 images from the right camera.

Figure 8. The edges detected in figure 6.

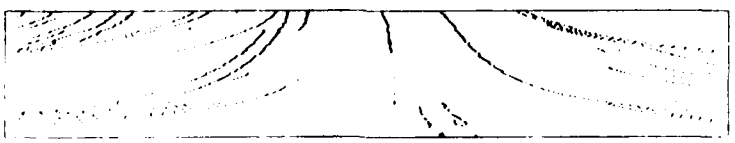


Figure 9. The edges detected in figure 7.

In the experiments reported in this paper we estimate $V\dot{R}$ at every pair of points on a trace separated by $3V$ steps in time, and use them to estimate C_E using equation (4) (by estimating $V\dot{R}$ rather than $\dot{R}$, only one division is necessary for each estimate of C_E). [1] Our estimates over all time and all traces are then averaged together. We choose $3V$ so that $\dot{R}$ will have had time to change significantly and render the estimate for C_E more stable.

[1] Thus each estimate relies on four points along the edge trace. Horn (1987) has suggested an iterative scheme for solving for a least squares fit along a complete trace.

A more sophisticated algorithm could dynamically choose the step size for the comparison based on the edge velocity.

Once C_E has been determined it is easy to compute the time to collision for a given edge point in a given edge using the same estimate for $\dot{R}$ as above. However since we are tracking edges and since edges correspond (hopefully) to a single three space feature, the time to collision should be reducing by one every image. We can therefore smooth our estimate for the time to collision by fitting a line of slope -1 over S images steps (i.e., $S + 1$ images). A least squares fit of such a line amounts to averaging the time to collision estimate along a trace over a set of $S + 1$ consecuctive images.

Thus our estimates for time to collision are valid for an image $(V + S)/2$ prior to the most recently processed image.

In all our experiments reported here we have used $S = 4$.

Using this algorithm the centers of expansion are estimated at pixels 243 and 274 in figures 8 and 9 respectively. These displays of edge traces are from a stereo pair of left and right cameras mounted on our mobile robot. The cameras are 576 pixels wide so these estimates say that each camera was skewed slightly to the right, with the left one skewed more than the right. This corresponds to our visual estimate from examining the camera mounts at the beginning of the experimental run. Figures 6 and 7 show the grey level images from which these edge arrays were extracted.

2.5 Approximation errors

It is worth asking how well this scheme does for computing depth even with synthetic data. There are a number of sources of inaccuracies; all edge locations are approximated to an integral number of pixels, divisions (such as in equation (4)) are rounded to the nearest integer, and the edge velocity estimates are only difference approximations to derivatives. With realistic camera parameters and using $V = 4$ and $S = 4$ we have experimentally found accuracy on synthetic edge arrays to be about 2%.

3. Calibrating Stereo

Recall that motion vision does not get good results close to the direction of motion since estimates for $\dot{R}$ are necessarily small and therefore susceptible to noise problems. Also it is useless when the robot is still, or not moving in straight lines. Stereo vision suffers from none of these problems. Our stereo algorithm matches edges in two one dimensional images. If all the camera parameters are known in advance we could then compute depth. An accurate model of the camera geometry is not sufficient because stereo is very susceptible to small camera misalignments and so the cameras must be repeatedly calibrated.

One approach to this problem is to run a calibration procedure prior to running the robot (e.g. Faugeras and Toscani (1986)). It usually involves placing the robot in front of a known test pattern and examining the test pattern with the visual system. It is usually necessary to run the calibration procedure before every run of the robot because of mechanical (thermal and other) drift in the relative position of the optics and the robot's drive mechanism.

A second approach is to calibrate from actual images during a run. Longuet-Higgins (1981) shows how to recover all camera parameters save a scale factor from a single pair of

images. However we also need to know the parameters of the cameras relative to the motion of the robot, as in general the camera platform can mechanically drift relative to the drive platform. Therefore we use the results of the forward motion algorithms to calibrate our cameras.

3.1 The algorithm

We use the same edge operator as for forward motion and apply the operator to each of the left and right images.

Matching of edges in the two images is accomplished by means of a dynamic programming algorithm similar to that developed by Ohta and Kanade (1986) for two dimensional images and by Serey and Matthies (1987) for single scanline images, except that the cost functions here are different.

The dynamic programming algorithm searches for a minimum cost path left to right across the two images where there is a cost associated with matching two edges, and a cost with skipping an edge whether it is due to noise, occlusion, uncovering or something else.

The skipping cost is a constant (2000 in the experiments reported in this paper).

The matching cost is a larger number (effectively ∞) unless the signs of the gradients of the two edges are the same (i.e., unless the grey levels of both edges go from dark to light or vice versa). Otherwise the matching cost is the sum of squares of differences of pixel grey levels on a small window around the edge (we used 7 pixels centered on the edge, in the experiments reported here).

3.2 What needs to be calibrated

There are a large number of parameters needed to describe the optical system used for stereo vision. Each of the two cameras has a focal ratio, and each has six positional and rotational degrees of freedom relative to the robot. We are forced to calibrate for some of these parameters, we can ignore some because their effects are too small to notice, and others we take care of by our choice of stereo algorithm, camera design, and methodology.

Figure 10 illustrates the camera geometry we are assuming. The robot coordinate system has the z-axis pointing in the direction of travel of the robot. The y-axis is vertical and the x axis is perpendicular to the direction of motion.

We assume that the two cameras have the same focal ratio f or field of view.[2] Our forward motion algorithm assumed some *a priori* knowledge of this parameter and we suggested a simple way to determine a rough estimate for it. Rather than rely on such an estimate we calibrate the stereo system assuming no previous knowledge of f.

Now consider the geometric degrees of freedom and refer to figure 10 for definitions of geometric quantities.

- x Our calibration assumes the base line separation of the cameras B in the x direction is unknown.
- y In general one would assume this parameter to be small, but in any case our use of the average of 16 lines of grey level data in the current experiments, and our

[2] It is possible to relax this assumption but the introduction of the necessary extra parameter in our calibration equations seems to make the calibration less stable. We do not have a theoretical basis for this remark--only empirical.

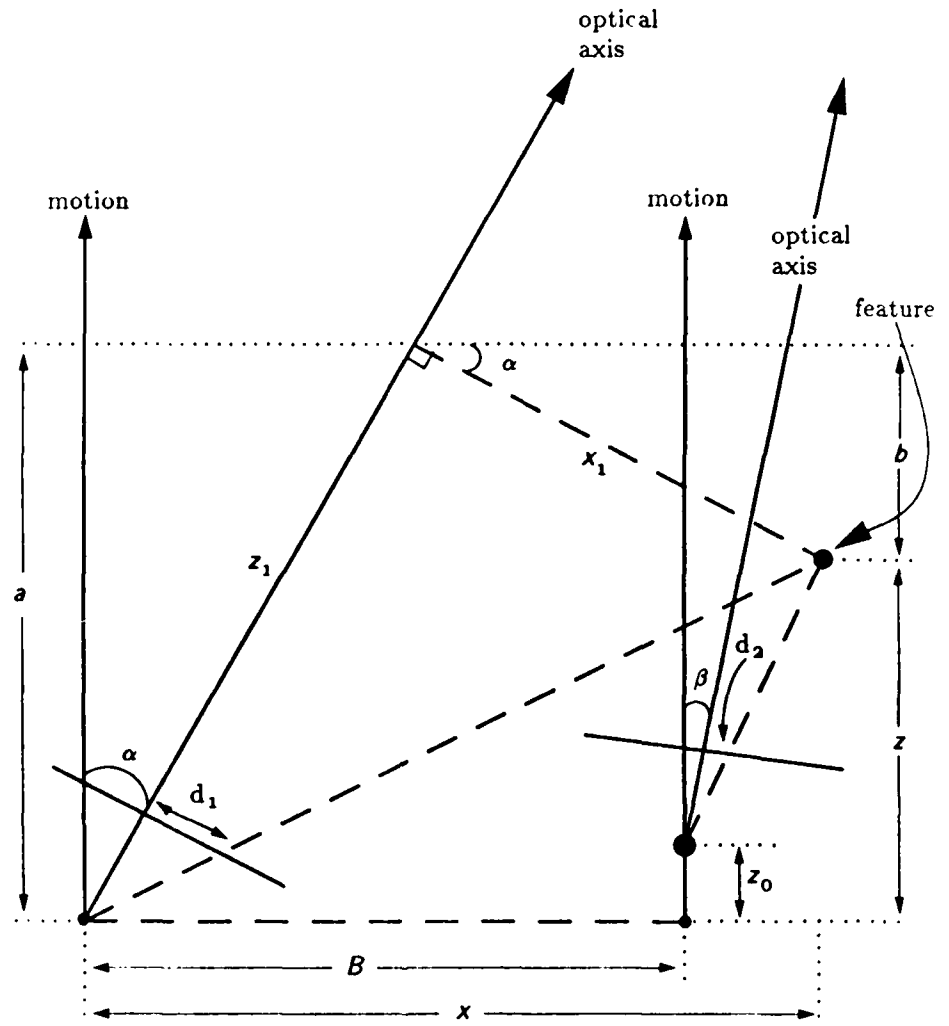


Figure 10. When there are two cameras they may both be misaligned relative to the direction of motion. They have a separation B perpendicular to the direction of motion and a misalignment z_0 in the direction of motion. Again we are interested in the forward distance z to some point.

intention to use cylindrical lenses mean that larger values of y will not effect the stereo measurements at all.

z There may be small errors in mounting the cameras which lead to a non-zero z_0. We show below that for small z_0 the effects are minimal and can be ignored.

pitch Again the averaging of scanlines or the use of cylindrical lenses takes care of this parameter.

roll One would expect this parameter to be small. One could measure the fuzziness of edges (since the image is really a vertically averaged, digitally or optically, image) to estimate the amount of roll present in the cameras. We will ignore this parameter.

yaw Each camera can swing from side to side on a standard camera mount, and in any case the orientation of the camera could easily drift mechanically from the

orientation of the robot's wheels. (It certainly does on our robots.) Our calibration assumes small unknown yaw angles on each camera.

3.3 Calibration formulation

Figure 10 shows a view from above of a stereo pair of cameras mounted on a mobile robot pointing approximately in the direction of motion. We assume both cameras have focal ratio f. The optical axis of the left camera is misaligned by angle α from the direction of motion, while the right camera is misaligned by angle β. The baseline separation, B, is measured perpendicular to the direction of motion. z_0 is the misalignment offset of the optical centers in the direction of motion.

We will assume that none of these five parameters (α, β, f, B, or z_0) is known *a priori*. We will, however, assume that we know bounds on some of these quantities. We assume that α and β are no bigger than$\pm 5°$, that f, in pixels, is comparable or larger than P the width of the image plane, (499 compared to 576 in our experiments) and that z_0 is much smaller than B.

The quantities we can directly measure given some matched feature in the two images are d_1 and d_2, the distances in pixels from the centers of view of the left and right cameras respectively. Due to our assumption above about the size of f, we know that roughly

$$|d_i| < \frac{f}{2}. \tag{8}$$

The quantity we wish to compute given some matched feature in the two images is its distance z in the direction of motion. Figure 10 shows that z is measured from the optical center of the left camera. The only other parameter we could compute is x, the displacement of the feature perpendicular to the direction of motion. We use x in deriving equations for z but do not compute it explicitly in any of our experiments. Given an estimate of f it can easily be recovered for use in obstacle avoidance.

Consider the quantities labelled a and b in the figure. They both relate to the z distance of the feature from the left camera. We can write

$$z = a - b$$

$$a = z_1 \cos\alpha$$

$$b = x_1 \sin\alpha$$

where z_1 is the distance of the feature in the direction of the optical axis of the left camera and x_1 is the feature's transverse distance from that axis. Now, because we are using perspective projection we can write

$$\frac{x_1}{z_1} = \frac{d_1}{f}$$

giving us an expression for x_1 in terms of z_1 and so we can rewrite z as

$$z = a - b = z_1(\cos\alpha - \frac{d_1}{f}\sin\alpha) \tag{9}$$

which is a linear expression in z_1. We can get a similar linear expression for z in terms of z_2 (it involves the constant offset z_0) and thus get a linear equation relating z_1 and z_2.

We can do the same analysis for x getting a second simultaneous linear equation relating z_1 and z_2. Solving and substituting back in equation (9) we obtain

$$z = \frac{\mu z_0 + \nu B}{\rho} \tag{10}$$

where

$$\mu = -fd_2 \cos\alpha \cos\beta + fd_1 \sin\alpha \sin\beta - f^2 \cos\alpha \sin\beta + d_1 d_2 \sin\alpha \cos\beta$$

$$\nu = f^2 \cos\alpha \cos\beta + d_1 d_2 \sin\alpha \sin\beta - fd_2 \cos\alpha \sin\beta - fd_1 \sin\alpha \cos\beta$$

$$\rho = (f^2 + d_1 d_2)\sin(\alpha - \beta) + f(d_1 - d_2)\cos(\alpha - \beta).$$

We now have an equation for z, the quantity we wish to compute, in terms of d_1 and d_2 the quantities we can measure. We want to derive a calibration procedure to identify the unknown parameters in this equation. We begin by noting that we can safely ignore some terms.

The sines of α and β are both less than 0.1. The cosines are all greater than 0.995. Thus the dominating term in μ is roughly fd_2 which by equation (8) is less than half the dominating term f^2 in ν. Since we also are assuming that z_0 is much smaller then B we will simply ignore the μ term. We can also ignore all but the f^2 term in ν as the other three are each at least 20 times smaller.

Turning attention to ρ we first observe that $(d_1 - d_2)$ can be arbitrarily small, and that $\sin(\alpha - \beta)$ can be larger than 0.17. Thus since either term can dominate both terms are important. The only remaining question is whether in the left hand term we can ignore $d_1 d_2$ as it is never more than 1/4 the size of f^2. We choose to ignore it for now in our derivation of a calibration procedure, but later we compare its inclusion and exclusion experimentally and conclude that it is insignificant.

With these approximations, and dividing both the top and bottom of equation (10) through by $f\cos(\alpha - \beta)$ we get

$$z = \frac{\Lambda}{\Gamma + d_1 - d_2} \tag{11}$$

where Λ and Γ are constants to be determined by calibration.

Suppose we have a collection of stereo images of features with known distance z, i.e., we have a collection of n triples (d_{1i}, d_{2i}, z_i). For a given Λ and Γ, we could write as a measure of closeness of fit for each triple

$$\frac{\Lambda}{z_i} - \Gamma - d_{1i} + d_{2i}. \tag{12}$$

This is both linear in Λ and Γ and a good measure of error in the image plane.

When we minimize the sum of squares of measure (12) for n calibration triples we obtain

$$\Lambda = \frac{n\sum(d_{1i} - d_{2i})/z_i - \sum(1/z_i)\sum(d_{1i} - d_{2i})}{n\sum(1/z_i^2) - (\sum 1/z_i)^2}$$

$$\Gamma = \frac{\sum(1/z_i)\sum(d_{1i} - d_{2i})/z_i - \sum(1/z_i^2)(\sum d_{1i} - d_{2i})}{n\sum(1/z_i^2) - (\sum 1/z_i)^2}$$

where all the sums range over the n values of i.

3.4 Calibration procedure

We collect triples to use for calibrating the stereo system by using forward motion analysis through equation (7) to get depth estimates for points visible in both the left camera and right camera. For each pair of images we know the pixel coordinates of each point for which

Figure 11. The left image of a stero pair at the beginning of a straight line motion of the robot.

we have a depth estimate. We run the stereo algorithm on each pair of images and for each match it finds we check to see whether forward motion delivered a depth estimate for both the edge in the left image and the edge in the right image. If so, and if those estimates are close (within $\pm 10\%$ of their average in our experiments) we use their average as the third element of a triple which includes the left and right edge pixel coordinates.

When a few tens of triples have been collected we plug them into the least squares formulation to get estimates of Λ and Γ.

3.5 Experimental results

To test the preceeding algorithms we set up our robot Allen to drive in an approximately straight line with large objects on either side of its path. Figures 11 and 12 are the left and right full frame camera views from approximately the start of its path.

The cameras were connected via cables to a Lisp machine and an image was digitized every 1/15 of a second giving a stereo pair every 2/15 of a second. The robot was moving at approximately 1.5 feet per second. Thus the contribution to z_0 (in figure 10) due to robot motion is approximately 1.2 inches, which is small compared to our stereo baseline B of approximately 8 inches and justifies our disregard of μ in equation (10).

Figures 6 and 7 show the left and right grey level motion images collected from 100 stereo pairs of standard images. Each scanline in figures 6 and 7 corresponds to averaging 16 scanlines from the centers of the original images. Time flows down these pseudo images, and is approximately 13 seconds from top to bottom. Figures 8 and 9 show the edges extracted from the time images --recall that the edge detection is one dimensional along scanlines.

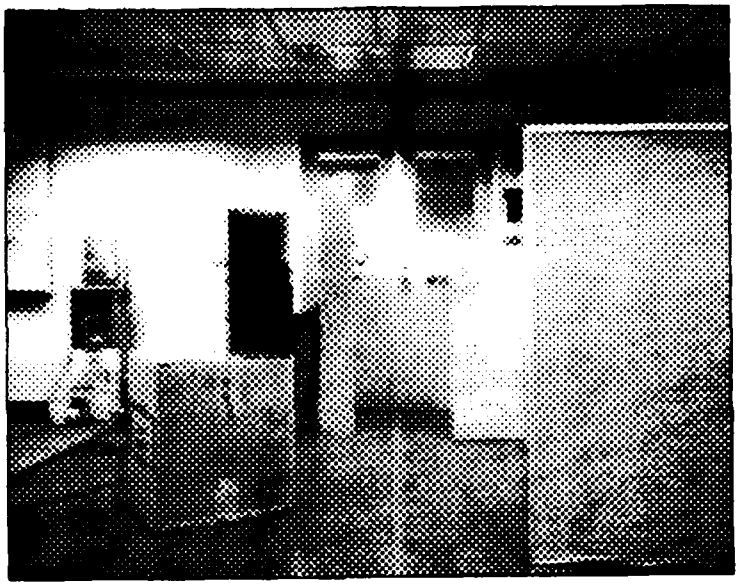


Figure 12. The right image of a stero pair at the beginning of a straight line motion of the robot.

In these experiments we made two passes over the edge images. The first was to estimate the centers of expansion and the second to extract motion depth estimates and correlate those with the stereo matches in order to get triples for stereo calibration. On a robot running these algorithms continuously there would only be one pass, simultaneously making minor refinements to both the centers of expansion and stereo calibrations.

In the 100 image pairs, the algorithm found 94 estimates for C_E in the left image, averaging to 243, and 110 estimates in the right image, averaging to 274. A total of 92 stereo matches were found, of which 31 were strongly consistent with motion depth estimates. These were used to calibrate the stereo system resulting in estimates of

$$\Lambda = 1958.8475, \quad \Gamma = 56.970116.$$

Figure 13 shows the original left and right averaged depth estimates from the motion algorithm for each of these 31 points, along with the estimate produced by the stereo algorithm with the derived calibration. The depth estimates are in units of time between collecting images. Notice that the difference between locations of matched edges in left and right images of particular edges is sometimes positive and sometimes negative because the cameras are not aligned to be parallel.

To test the stereo calibration we took four feature points corresponding to known objects in the scene. We estimated where the robot had been when the first images in the sequence were taken (to achieve the constant velocity constraint the robot had to be moving before we started collecting images) and measured the distances to the known points in the direction of motion. We substituted the left and right image edge coordinates into equation (12) to get time to collision estimates, and divided each of those into the known distances (measured in feet) and averaged the result to get a calibration of the stereo system in feet. The estimate is 0.1874 feet per stereo pair image step, giving the robot's velocity at 1.406 feet per second. Figure 14 shows the unsurprising results of this procedure in the top four rows of the table.

l	r	disp	motion	stereo	l	r	disp	motion	stereo
144	163	−19	50	52	163	189	−26	69	63
525	550	−25	60	61	521	545	−24	60	59
169	197	−28	54	68	516	541	−25	60	61
170	198	−28	62	68	512	537	−25	60	61
507	532	−25	60	61	172	202	−30	71	73
503	530	−27	60	65	174	203	−29	69	70
174	204	−30	74	73	149	137	12	28	28
175	205	−30	63	73	153	142	11	28	29
155	146	9	30	30	73	68	5	32	32
77	73	4	32	32	81	79	2	33	33
85	85	0	34	34	90	90	0	36	34
93	96	−3	36	36	102	106	−4	36	37
105	111	−6	36	38	9	48	−39	112	109
11	51	−40	115	115	435	441	−6	46	38
13	54	−41	122	123	430	439	−9	48	41
426	435	−9	49	41					

Figure 13. Display of the stereo calibration and results. Left two columns are pixel coordinates of edges in the left and right images, third column is their difference, fourth column is the average depth estimate from the motion algorithms running on the left and right image sequences and the fifth column is the depth estimate produced by the stereo system calibrated with this data.

We then took the same four features and searched for them 20 images later in the image sequences. The bottom half of figure 14 shows the stereo estimates for these four points. Ideally, all the time to collision estimates should be reduced by 20. They are reduced by 18, 14, 20, and 17. Note that the third of these points appeared roughly in the centers of the two images and no time to collision estimates were produced for it by the motion algorithms. The fourth column shows the distance in feet that would have been estimated if the stereo had come up with a decrease of exactly 20 units for each of the points. The sixth column shows the estimate that was actually obtained. The relative distances between the points known a priori from measurement given in the fourth column in the upper part of the table, and the estimated distances in the sixth column of the lower part of the table are quite good and can be summarized as:

exact	—	est
5.0	—	5.6
8.6	—	7.5
2.0	—	1.3
3.6	—	1.9
7.0	—	6.9
10.6	—	8.8

Note that these are relative distances, some of them at large distances from the robot itself.

l	r	disp	known	ttc	est
194	216	−22	10.5	56	10.5
314	347	−33	15.5	82	15.4
263	300	−37	19.1	98	18.4
398	414	−16	8.5	48	9.0
173	179	−6	6.7	38	7.1
336	364	−28	11.6	68	12.7
273	305	−32	14.6	78	14.6
485	479	6	5.2	31	5.8

Figure 14. The first four rows show features with known distances (fourth column) in feet, their depth (fifth column) in time to collision units as delivered by the calibrated stereo algorithm, and their depth in feet (sixth column) by calibrating the previous two columns. In the second four rows, the same points are displayed from images 20 time units later. This time the fourth column is the predicted distance based on the sixth column above and the sixth column is the estimate from stereo.

l	r	disp	known	ttc	est
194	216	−22	10.5	56	10.5
314	347	−33	15.5	82	15.3
263	300	−37	19.1	99	18.5
398	414	−16	8.5	48	9.0
173	179	−6	6.7	38	7.1
336	364	−28	11.6	68	12.7
273	305	−32	14.8	79	14.8
485	479	6	5.2	31	5.8

Figure 15. Same as for figure 14 but using a much more complex stereo calibration procedure. There is almost no difference in results.

3.6 Other calibration schemes

In experiments with synthetic data the most signifcant term omitted from equation (10) in equation (11) appeared to be $d_1 d_2$ in the denominator. We therefore repeated the above experiment using a calibration equation of the form

$$z = \frac{\Lambda}{\Gamma + \Omega d_1 d_2 + d_1 - d_2}. \tag{13}$$

The results were almost identical:

$$\Lambda = 1957.6881, \quad \Gamma = 56.864132, \quad \Omega = 2.360905 \times 10^{-6},$$

an estimate of 0.1869 feet per image step (within 0.3% of the previous estimate) or robot velocity of 1.401 feet per second, and almost identical results on the check with known distances as shown in figure 15.

These experiments convinced us that (11) is certainly sufficient. But is it necessary? To check we tried calibrating with the same data to a model of stereo that assumes the cameras

l	r	disp	known	ttc	est
194	216	−22	10.5	55	18.0
314	347	−33	15.5	37	12.1
263	300	−37	19.1	33	10.8
398	414	−16	8.5	76	24.8
173	179	−6	11.4	202	66.0
336	364	−28	5.6	43	14.0
273	305	−32	4.2	38	12.4
485	479	6	18.3	−202	−66.0

Figure 16. Same as for figure 14 but using a stereo calibration that assumes the cameras are aligned parallel to the direction of motion. The results are meaningless.

are perfectly aligned, i.e., using a calibration equation of the form:

$$z = \frac{\Lambda}{d_1 - d_2}. \tag{14}$$

The results were:

$$\Lambda = -1213.2903,$$

an estimate of 0.3266 feet per image step or robot velocity of 2.450 (almost a factor of 2 off), and meaningless results on known distances as shown in 16. Clearly this naive model is not sufficient.

4. Conclusions

In this paper we have demonstrated a number of aspects of forward motion analysis and stereo vision.

The results of forward motion analysis are fairly noisy by the unnatural standards set by most workers in computer vision and mobile robots. In this paper we first argued that those standards are not necessary, and in fact are not met by humans, most of whom operate perfectly well as autonomous mobile agents. In fact errors of $\pm 10\%$ do not seem to large to us for reliable mobile robot obstacle avoidance.

Forward motion analysis has some wonderful independence properties and only requires a small number of 16 bit fixed precision arithmetic operations to deliver depth in a coordinate system natural to the task of obstacle avoidance. At the same time as it is delivering depth estimates it can continually recalibrate itself for camera misalignment relative to the direction of motion. All of these computations could easily be transfered to a parallel network of simple processors.

Despite the local noise in each forward motion estimate we demonstrated that with just a few tens of such measurements we could calibrate a stereo camera system with grossly misaligned cameras.

What does this buy us?

It means we can have a vision system on board a robot that doesn't require a time consuming calibration stage at the beginning of an experimental run. Rather we can build power-up-and go systems. If the sensor platform drifts mechanically from the drive alignment (as it does on many real robots) over time there is no calibration problem—the al-

gorithms given in this paper continually adjust. If there is more severe and sudden misalignment, e.g., due to a hard collision, the robot will be disoriented for only a few seconds before it accommodates.

More than this however, the possibility of simple fast dynamic calibration to the world opens up the possibility of having cheap (and hence mechanically sloppy) steerable sensor platforms. If we can quickly and cheaply compute all we need to know about how such a platform is aligned we will not feel pressure to spend inordinate amounts of money building a precise platform.

Silicon is getting cheaper a lot quicker than precision machined parts are. We should search for ways, as this paper does, of trading off silicon for such precision machined parts. If the analysis and algorithms are simple we have a better chance of them being robust. Of course it is critical to back up analysis with experiment—analysis in computer vision without experiment is often a worthless intellectual pursuit.

Acknowledgements

Claudia Smith digitally drew the figures for this paper. Chris Lindblad and Dave Clemens coaxed digital images from laser printers. Berthold K. P. Horn has provided many helpful comments on earlier drafts of this paper which have improved it greatly.

References

Bolles, Baker and Marimont (1987) "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion", Robert C. Bolles, H. Harlyn Baker and David H. Marimont, *International Journal of Computer Vision*, 1, 7–55.

Brooks (1986) "A Robust Layered Control System for a Mobile Robot", Rodney A. Brooks, *IEEE Journal of Robotics and Automation*, RA-2, April, 14–23.

Brooks (1987) "A Hardware Retargetable Distributed Layered Architecture for Mobile Robot Control", Rodney Brooks, *Proceedings IEEE Robotics and Automation, Raleigh NC*, April, 106–110.

Longuet-Higgins (1981) "A Computer Algorithm for Reconstructing a Scene from Two Projections", H. C. Longuet-Higgins, *Nature*, 293, 133–135.

Faugeras and Toscani (1986) "The Calibration Problem for Stereo", Olivier D. Faugeras and G. Toscani, *Proceedings of Computer Vision and Pattern Recognition*, Miami Beach, 15–20.

Horn (1986) "Robot Vision", Berthold K. P. Horn, *MIT Press, Cambridge.*

Horn (1987) *personal communication.*

Marr (1982) "Vision", David Marr, *W. H. Freeman, San Francisco.*

Ohta and Kanade (1983) "Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming", Yuichi Ohta and Takeo Kanade, *CMU Tech Report.*

Serey and Matthies (1987) "Obstacle Avoidance Using 1-D Stereo Vision", Bruno Serey and Larry Matthies, *CMU Tech Report*, in preparation.

Appendix—Kludge Factors

Almost all computer vision programs have a large number of "tweakable" parameters hidden in their code. These are usually used to implement certain English phrases used in the scientific paper which describes the work. For instance there might be a phrase like "for sufficiently close edge terminations we ...". In this paper we have tried to explicity state all such *kludge factors* that we have used. For the reader's convenience we also collect them here and give the values we used for them throughout all experiments we have done.

The edge mask (derivative of a Gaussian) is:

$$1\ 3\ 5\ 9\ 14\ 18\ 20\ 18\ 11\ 0\ -11\ -18\ -20\ -18\ -14\ -9\ -5\ -3\ -1.$$

The threshold strength we demand of edges is 500. The same edges are used for stereo and motion algorithms.

In the dynamic programming portion of the stereo algorithm we assign a cost of 2000 to skipped matches, and the sum of squares of pixel grey level differences in a window of 7 pixels centered on the left and right edges for accepted matches.

In linking edges in the motion algorithm we start off with a search window of ± 3 pixels and later narrow that to -1 to $+3$ pixels from the most recent step.

To compute the edge velocity in the motion algorithms we use the slope of a chord joining edge locations $V = 4$ images apart. In estimating C_E we do this twice at $3V = 12$ images apart.

We smooth the time to collision estimates over $S = 4$ time intervals (i.e., 5 images).

In deciding on consistent depth estimates and stereo matches we demand that the left and right motion depth estimates lie within $\pm 10\%$ of their average.

END

12-87

DTIC